SlaveResume

After a slave DFO sends a ready message, it waits for a resume message from the QC to continue processing. When it receives a resume message, a slave DFO resumes its execution. FIG. 7E illustrates a SlaveResume process flow. At decision block **750** (i.e., "first row already fetched?"), if the first row has already been fetched, processing continues at block **752** to write the first row to the slave DFO's output TQ, and processing continues at block **754**. If the first row has not been fetch, processing continues at block **754**. At block **754**, SlaveFetch is invoked to fetch any remaining rows from the DFO cursor. Processing ends at block **756**.

SlaveClose

Upon completion of a DPO, the database cursor associated with the completed DFO can be closed. FIG. 7E illustrates a process flow for SlaveClose. At block **762**, the DFO's database cursor is closed. Processing ends at block **764**.

SlaveFetch

If a "ready" reply is not expected, or a slave DFO receives a resume after sending a "ready" reply, a slave DFO can fetch all the rows from a DFO cursor. FIG. 7F illustrates a process flow to fetch all rows. At decision block **770** (i.e., "given DFO cursor is at EOF?"), if the DFO cursor is at eof, processing continues at decision block **772**.

At decision block **772** (i.e., "QC expects 'partial' reply?"), if a partial execution message is expected by the QC to indicate that the slave DFO has completed processing the range of rowids provided by the QC, processing continues at block **774** to send the partial message to the QC, and processing ends at **780**. If a partial message is not expected (as determined at decision block **772**), processing continues at block **778** to write an eof to the output TQ for the slave DFO. Processing ends at block **780**.

If, at decision block **770** (i.e., given DFO cursor is at EOF?"), it is determined that the DFO cursor is not at eof, processing continues at block **776** to fetch the next row from the DFO cursor. At block **782**, the row is written to the slave DFO's output TQ, and processing continues at decision block **770** to fetch any remaining rows from the DFO cursor.

OTHER PARALLELIZATION EXAMPLES

One application for parallelism is the creation of an index. The index creation operation includes table scan, subindex create, and merge subindices operations. The table scan and subindex create can be performed in parallel. The output from the subindex creation operation is the input to the merge subindices process. In addition, a table can be created in parallel. A subtable create operation can create a table in parallel, and the subtables can be merged.

Thus, a method and apparatus for processing queries in parallel has been provided.

We claim:

1. A computer-implemented method of implementing database management system (DBMS) operations in parallel, independent of physical storage locations, said computer-implemented method comprising the steps of:

generating a serial execution plan for operations in said DBMS;

generating a parallelized execution plan for said serial execution plan, said parallelized execution plan including first and second operations, said second operation including one or more slave processes operating on a plurality of data partitions, the quantity of said data partitions being greater than the quantity of said slave processes, each of said slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

executing said parallelized execution plan when a plurality of parallel resources of said computer system are available, said first and second operations executing in parallel; and

executing said serial execution plan when said plurality of resources are not available.

2. The method of claim 1 wherein said step of generating a parallelized execution plan includes the steps of:

identifying one or more segments of said serial execution plan that can be parallelized; and

identifying partitioning requirements of said one or more segments.

3. The method of claim 1 wherein said step of generating a parallelized execution plan is based on a specification of parallelism in a statement specifying one of said operations.

4. A method of generating an execution plan to process a database management system (DBMS) operation in parallel including the steps of:

generating an execution plan for said operation;

examining said execution plan from bottom up;

identifying a parallelized portion of said execution plan, said parallelized portion can be processed in parallel, said parallelized portion including first and second operations, said first and second operations being executable in parallel, said second operation including one or more slave processes operating on a plurality of data partitions, the quantity of said data partitions being greater than the quantity of said slave processes, each of said slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

identifying some serial portion of said execution plan, said serial portion can be processed in serial;

allocating a central scheduler between said parallelized portion and said serial portion.

5. The method of claim 4 further including the steps of:

identifying a first data flow requirement for a first portion of said execution plan said first data flow requirement corresponding to a partitioning of a data flow required by said first portion;

identifying a second data flow requirement for a second portion of said execution plan said second data flow requirement corresponding by said second portion; and

allocating a data flow director between said first portion and said second portion when said first data flow requirement is not compatible with said second data flow requirement said data flow director repartitioning a data flow of said first portion to be compatible with said second data flow requirement.

6. A computer-implemented method of executing database management system (DBMS) operations in parallel in a computer system, said method comprising the steps of:

generating an execution plan to execute said operations in parallel, said execution plan including first and second operations;

initiating an operation coordinator in said computer system to coordinate execution of said execution plan;

initiating, by said operation coordinator, a first set of slaves operating on a plurality of data partitions to produce data, the quantity of said data partitions being greater than the quantity of said first set of slave processes, each of said slave processes of said first set of slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

initiating, by said operation coordinator, a second set of slaves to consume data; and

directing said second set of slaves to produce data and said first set of slaves to consume data when said first set of slaves finishes producing data.

7. The method of claim 6 wherein said execution plan is comprised of operator nodes and said operator nodes are linked together to form execution sets.

8. A computer-implemented method of executing database management system (DBMS) operations in parallel in a computer system, said method comprising the steps of:

generating an execution plan to execute said operations in parallel, said execution plan including first and second operations;

initiating a data flow scheduler in said computer system to coordinate data flow;

initiating, by said data flow scheduler, producer slaves operating on a plurality of data partitions to produce a first data production the quantity of said data partitions being greater than the quantity of said producer slaves, each of said producer slaves operating on a different one of said data partitions, at least one of said producer slaves operating on more than one of said data partitions;

initiating, by said data flow scheduler, consumer slaves to consume said first data production;

transmitting a ready message to said data flow scheduler when said producer slaves become ready to produce data;

transmitting a completion message to said data flow scheduler when said first data production is completed:

generating, by said data flow scheduler, in response to said completion message, an identification of a plurality of said consumer slaves that did not receive data in said first data production, said generating step using information derived from said ready message;

examining, by said producer slaves, said identification during a subsequent data production; and

reducing said subsequent data production such that said subsequent data production does not produce data for said plurality of said consumer slaves.

9. In a computer system, a database management apparatus for implementing database operations in parallel, independent of physical storage locations, said database management apparatus comprising;

means for generating a serial execution plan for operations in said database management apparatus;

means for generating a parallelized execution plan for said serial execution plan, said parallelized execution plan including first and second operations, said second operation including one or more slave processes operating on a plurality of data partitions, the quantity of said data partitions being greater than the quantity of said slave processes, each of said slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

means for executing said parallelized execution plan when a plurality of parallel resources of said computer system are available, said first and second operations executing in parallel; and

means for executing said serial execution plan when said plurality of resources are not available.

10. The apparatus of claim 9 wherein said means for generating a parallelized execution plan further includes:

means for identifying one or more segments of said serial execution plan that can be parallelized; and

means for identifying partitioning requirements of said one or more segments.

11. The apparatus of claim 9 wherein said means for generating a parallelized execution plan is based on a specification of parallelism in a statement specifying one of said operations.

12. In a computer system, a database management apparatus for generating an execution plan to process database operations in parallel, said database management apparatus comprising;

means for generating an execution plan for said operations;

means for examining said execution plan from bottom up;

means for identifying a parallelized portion of said execution plan, said parallelized portion including first and second operations, said parallelized portion being processed in parallel, said second operation including one or more slave processes operating on a plurality of data partitions, the quantity of said data partitions being greater than the quantity of said slave processes, each of said slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

means for identifying some serial portion of said execution plan, said serial portion being processed in serial; and

means for allocating a central scheduler between said parallelized portion and said serial portion.

13. The apparatus of claim 12 further including:

means for identifying a first data flow requirement for a first portion of said execution plan said first data flow requirement corresponding to a partitioning of a data flow required by said first portion;

means for identifying a second data flow requirement for a second portion of said execution plan said second data flow requirement corresponding to said second portion; and

means for allocating a data flow director between said first portion and said second portion when said first data flow requirement is not compatible with said second data flow requirement, said data flow director repartitioning a data flow of said first portion to be compatible with said second data flow requirement.

14. In a computer system, a database management apparatus for executing database operations in parallel, said database management apparatus comprising:

means for generating an execution plan to execute said operations in parallel, said execution plan including first and second operations, said first and second operations being executed in parallel;

means for initiating an operation coordinator in said computer system to coordinate execution of said execution plan;

means for initiating, by said operation coordinator, a first set of slaves operating on a plurality of data partitions to produce data, the quantity of said data partitions being greater than the quantity of said first set of slave processes, each of said slave processes of said first set of slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

means for initiating, by said operation coordinator, a second set of slaves to consume data; and

means for directing said second set of slaves to produce data and said first set of slaves to consume data when said first set of slaves finishes producing data.

15. The apparatus of claim 14 wherein said execution plan is further comprised of operator nodes and said operator nodes are linked together to form execution sets.

16. In a computer system, a database management apparatus for executing database operations in parallel, said database management apparatus comprising:

means for generating an execution plan to execute said operations in parallel, said execution plan including first and second operations, said first and second operations being executed in parallel;

means for initiating a data flow scheduler in said computer system to coordinate data flow;

means for initiating, by said data flow scheduler, producer slaves operating on a plurality of data partitions to produce a first data production, the quantity of said data partitions being greater than the quantity of said producer slaves, each of said producer slaves operating on a different one of said data partitions, at least one of said producer slaves operating on more than one of said data partitions;

means for initiating, by said data flow scheduler, consumer slaves to consume said first data production;

means for transmitting a ready message to said data flow scheduler when said producer slaves become ready to produce data;

means for transmitting a completion message to said data flow scheduler when said first data production is completed;

means for generating, by said data flow scheduler, in response to said completion message, an identification of a plurality of said consumer slaves that did not receive data in said first data production, said generating step using information derived from said ready message;

means for examining, by said producer slaves, said identification during a subsequent data production; and

means for reducing said subsequent data production such that said subsequent data production does not produce data for said plurality of said consumer slaves.

17. An article of manufacture comprising a computer usable mass storage medium having computer readable program code embodied therein for causing a processing means to execute computer-implemented database management operations in parallel, independent of physical storage locations, said computer readable program code in said article of manufacture comprising:

computer readable program code for causing said processing means to generate a serial execution plan for said database management operations;

computer readable program code for causing said processing means to generate a parallelized execution plan for said serial execution plan, said parallelized execution plan including first and second operations, said second operation including one or more slave processes operating on a plurality of data partitions, the quantity of said data partitions being greater than the quantity of said slave processes, each of said slave processes operating on a different one of said data partitions, at least one of said slave processes operating on more than one of said data partitions;

computer readable program code for causing said processing means to execute said parallelized execution plan when a plurality of parallel resources of said computer system are available, said first and second operations executing in parallel; and

computer readable program code for causing said processing means to execute said serial execution plan when said plurality of resources are not available.

18. The article of manufacture of claim 17 wherein said computer readable program code for causing said processing means to generate a parallelized execution plan further includes:

computer readable program code for causing said processing means to identify one or more segments of said serial execution plan that can be parallelized; and

computer readable program code for causing said processing means to identify partitioning requirements of said one or more segments.

19. The article of manufacture of claim 17 wherein said computer readable program code for causing said processing means to generate a parallelized execution plan is based on a specification of parallelism in a statement specifying one of said operations.

* * * * *

Add
a'